Lecture 26: Online Algorithms

12/05/2020

1 Introduction: Stairs vs. Elevators Problem

Consider the "Stairs vs. Elevators" problem. Let:

- S: Time to take stairs.
- L: Time to take elevator.
- Assume $S > L$.

Problem: How long should one wait for the elevator? If one knows the distribution of the arrival time of the elevator, then one can compute various quantities. We will work with the worst-case model, which has merits and cons. The **Adversary** decides the arrival time.

2 Deterministic Strategy

Is there a good deterministic strategy? Let $T = S - L$.

Let $OPT(t)$ be the optimum total time if the elevator arrives at time t.

- If $t \leq T$, then $OPT(t) = t + L$ (Wait for elevator).
- If $t > T$, then $OPT(t) = S$ (Take stairs immediately/early).

For an algorithm A, let $A(t)$ be the total time if the elevator arrives at time t. The **Competitive Ratio** of algorithm A is:

$$C(A) = \sup_t \frac{A(t)}{OPT(t)} \tag{1}$$

2.1 Characterizing Deterministic Algorithms

Any deterministic algorithm for this problem can be characterized by a single parameter a: the time it will wait before taking the stairs. (It will take the elevator if it comes before a).

Let $Wait_a$ be the algorithm. What is its competitive ratio?

Case $Wait_0$: This strategy is to always take the stairs. If $t = 0^+$ (elevator arrives immediately), then $Wait_0$ takes S time, while $OPT(0^+) = L$. The competitive ratio is $\frac{S}{L}$, which can be arbitrarily large.

General $Wait_a$: One can see that the worst case for $Wait_a$ is if the elevator comes at time $a + \epsilon$ (just after we give up and take the stairs). In this case, the competitive ratio is:

$$\frac{a+S}{\min(a+L, S)} \tag{2}$$

It is not hard to see that $a = S - L = T$ is the minimizing value (intuitively, it does not make sense to wait longer than T). Substituting $a = S - L$:

$$\text{Competitive Ratio} = \frac{S - L + S}{S} = 2 - \frac{L}{S} \tag{3}$$

Theorem: The optimum competitive ratio for the deterministic problem is $2 - \frac{L}{S}$.

3 Randomized Algorithms

What if we allow randomization in the algorithm? Can we improve the ratio?

3.1 The Model

Need to be careful about the model. We will consider an **Oblivious Adversary** that cannot see the randomness of the algorithm.

Alternatively, we let the adversary know the algorithm, pick the worst-case input σ, and then we allow the algorithm to randomize. We define the competitive ratio as:

$$\max_{\sigma} \frac{E[A(\sigma)]}{OPT(\sigma)} \tag{4}$$

Note that $A(\sigma)$ is a random variable.

3.2 Optimal Randomized Strategy

For the elevator problem, we view a randomized algorithm as picking the waiting time according to a distribution. What distribution? It turns out the optimal distribution is to pick $w \in [0, T]$ according to the density function:

$$p(t) = \frac{1}{(e-1)T} e^{t/T} \tag{5}$$

Note that $\int_0^T p(t)dt = 1$. And $p(t) = 0$ for $t > T$.

Theorem: The competitive ratio of the randomized algorithm that picks waiting time according to $p(t)$ is $\frac{e}{e-1} \approx 1.58$.

3.3 Proof Sketch

Fix any arrival time $a \in [0, T]$. Suppose the elevator arrives at time a. Then $OPT(a) = a + L$.

What about the algorithm? Its total time is:

- $a + L$ if the waiting time picked is $\geq a$.

- $t + S$ if the waiting time picked is $t < a$.

Thus, the expected cost is:

$$E[A(a)] = \int_0^a (S+t)p(t)dt + \left(\int_a^T p(t)dt\right)[a+L] \tag{6}$$

The worst case happens when $a = T$ (as in the deterministic case). Then we have:

$$\begin{aligned} E[A(T)] &= S + \int_0^T tp(t)dt \\ &= S + \int_0^T \frac{t}{(e-1)T} e^{t/T} dt \end{aligned}$$

Using integration by parts:

$$\begin{aligned} &= S + \frac{1}{(e-1)T}\left[Tte^{t/T} - T^2 e^{t/T}\right]_0^T \\ &= S + \frac{1}{e-1}T \end{aligned}$$

While $OPT(T) = S$.

Hence, the competitive ratio is:

$$\frac{S + \frac{1}{e-1}T}{S} = \frac{S + \frac{S-L}{e-1}}{S} = 1 + \frac{1}{e-1} - \frac{L}{S(e-1)} \tag{7}$$

If we let $L/S \to 0$, the ratio tends to:

$$1 + \frac{1}{e-1} = \frac{e-1+1}{e-1} = \frac{e}{e-1} \tag{8}$$

4 Lower Bounds: Yao's Min-Max Principle

How do we prove that the above ratio is optimal? In general, how do we prove lower bounds on randomized algorithms?

Typically, we use what is called **Yao's Lemma**. Yao recognized early on that one can use the Von-Neumann Min-Max characterization of two-player games to prove lower bounds.

4.1 The Principle

Instead of explaining the general setup, we illustrate it in the context of this discrete problem. To prove a lower bound on randomized algorithms, we focus on a lower bound on **deterministic algorithms against inputs drawn from a distribution**.

The game is as follows:

1. Come up with a probability distribution on inputs, say D.
2. Now, given knowledge of D, what is the best deterministic algorithm?
3. Suppose no deterministic algorithm can do better than C (even with knowledge of D).
4. Then C is a lower bound on randomized algorithms.

4.2 Application to Elevator Problem

For our problem, we design/pick a distribution on the arrival time of the elevator. Suppose we pick the distribution e^{-t} for the elevator arrival time. Note that we are allowing arbitrarily large times, but that is ok. To simplify analysis we assume $S = 1, L = 0$.

What is the expected value of OPT?

$$E[OPT] = \int_0^\infty \min(1,t)p(t)dt = \int_0^1 te^{-t}dt + \int_1^\infty 1e^{-t}dt = 1 - e^{-1} \tag{9}$$

Fix any deterministic algorithm. It is a threshold algorithm with parameter a. Its expected cost is:

$$E[A(a)] = \int_0^a te^{-t}dt + \int_a^\infty (a+1)e^{-t}dt = 1 \tag{10}$$

(Result doesn't depend on the threshold a!).

Thus, the competitive ratio is:

$$\geq \frac{1}{1-\frac{1}{e}} = \frac{e}{e-1} \tag{11}$$

5 Ski Rental Problem

A closely related problem, but in some sense a discrete problem. More well known.
Setup: Costs \$$B$ to buy skis and \$1 to rent for 1 day ($B > 1$). Every day one has to decide whether to rent or buy if not already bought. Adversary decides when to break one's leg and finish the ski season. What is the best strategy to minimize total cost?

Deterministic: Rent for $\lfloor B \rfloor$ days and then buy. One can show 2-competitive and optimum for deterministic. Can obtain $(1-\frac{1}{e})^{-1}$ for randomized. Somewhat more technical than elevator/stairs problem because of discrete days.

6 Yao's Min-Max Principle (Formal View)

Consider a discrete setting where we enumerate all inputs of a particular size and all deterministic algorithms. Say m algorithms ($\mathcal{A}_i$) and n inputs (I_j).

	I_1	$\dots$	I_n
$\mathcal{A}_1$	c_{11}	$\dots$	c_{1n}
$\vdots$		$\ddots$	
$\mathcal{A}_m$	c_{m1}	$\dots$	c_{mn}

Let p be a distribution over inputs. Let q be a distribution over algorithms.

Let q^* be an optimal randomized algorithm. This corresponds to choosing q^* to minimize:

$$\max_{I_j} \frac{\sum_{i=1}^m q_i^* A_i(I_j)}{OPT(I_j)} = c^* \tag{12}$$

By Min-Max Theorem (swapping min and max over distributions):

$$\begin{aligned}\min_q \max_p E[\text{cost}] &= \max_p \min_q E[\text{cost}] \\ &= \min_{A_i} \sum_{j=1}^n p_j \frac{A_i(I_j)}{OPT(I_j)}\end{aligned}$$

7 Probabilistic Tree Embedding

Let $G = (V, E)$ be a graph. Want to approximate distances in G by a tree. Why? Trees are simple. Can one approximate distances by a single spanning tree?

Consider an n-cycle. For any spanning tree T, there is an edge $e = uv$ removed. Then $d_T(u, v) = n - 1$, while $d_G(u, v) = 1$. This is a large distortion.

Can we use randomization to improve this? Yes!

Theorem: There exists a probability distribution p over spanning trees $ST(G)$ such that for all u, v:

$$E_{T\sim p}[d_T(u, v)] \leq O(\log n) d_G(u, v) \tag{13}$$

Moreover, one can sample a tree from p efficiently.

Many applications to algorithms. The bound $O(\log n)$ can be improved to $O(\log n)$ if G is a complete graph that corresponds to a metric space. Lower bound is $\Omega(\log n)$ even for planar graphs (or general graphs as per context).